# National Textile University

## Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

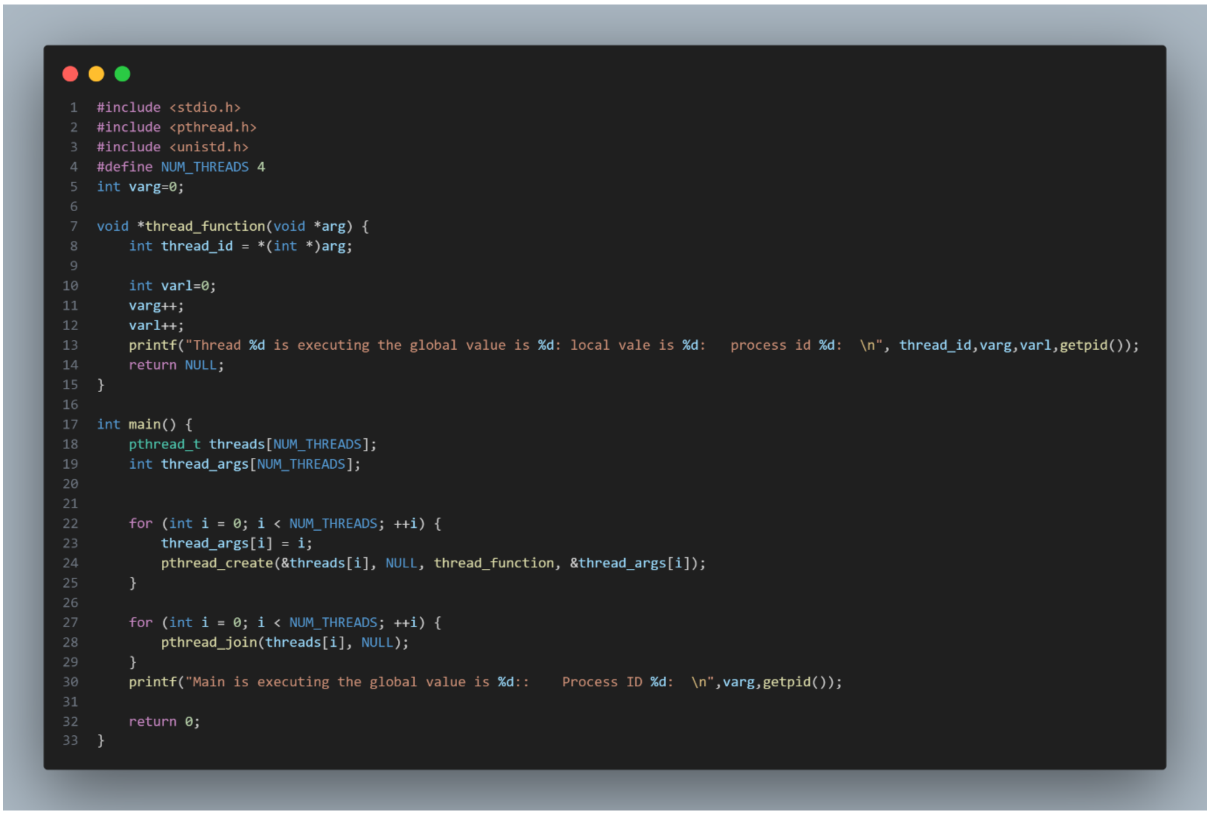
Semester:

5th- A

# LAB-6

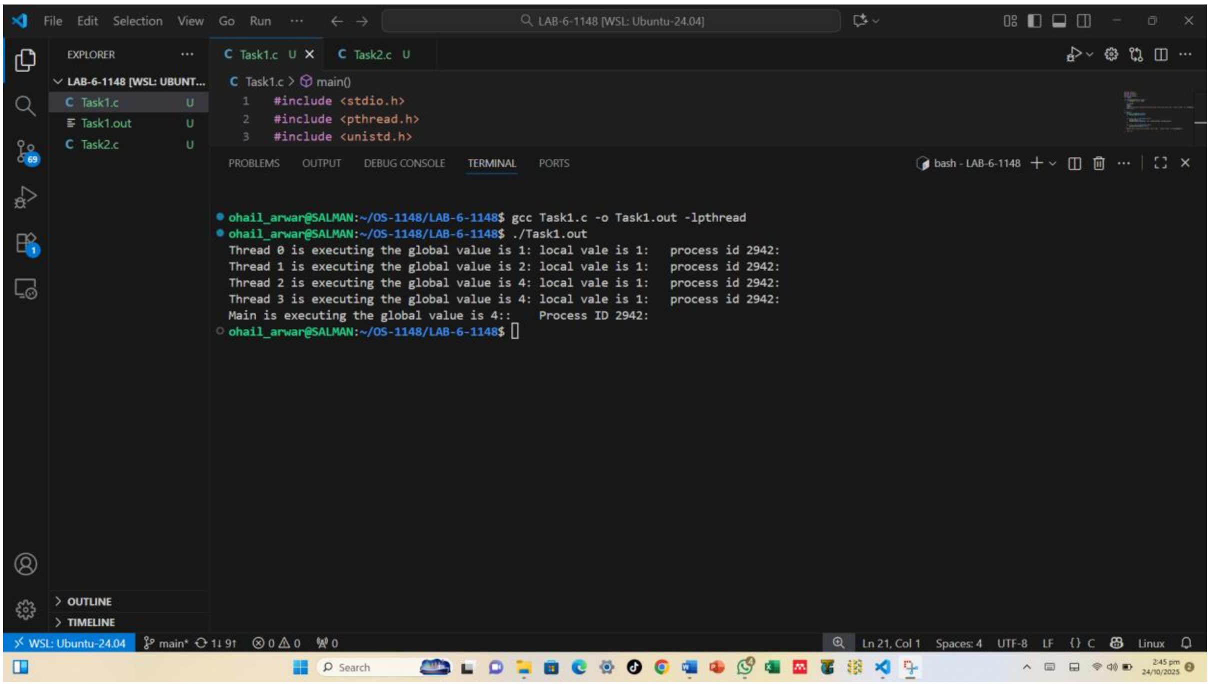
## TASK-1

### Code:



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_THREADS 4
int varg=0;

void *thread_function(void *arg) {
    int thread_id = *(int *)arg;

    int varl=0;
    varg++;
    varl++;
    printf("Thread %d is executing the global value is %d: local vale is %d:   process id %d:  \n", thread_id,varg,varl,getpid());
    return NULL;
}

int main() {
    pthread_t threads[NUM_THREADS];
    int thread_args[NUM_THREADS];


    for (int i = 0; i < NUM_THREADS; ++i) {
        thread_args[i] = i;
        pthread_create(&threads[i], NULL, thread_function, &thread_args[i]);
    }

    for (int i = 0; i < NUM_THREADS; ++i) {
        pthread_join(threads[i], NULL);
    }
    printf("Main is executing the global value is %d::    Process ID %d:  \n",varg,getpid());

    return 0;
}
```

### Output:

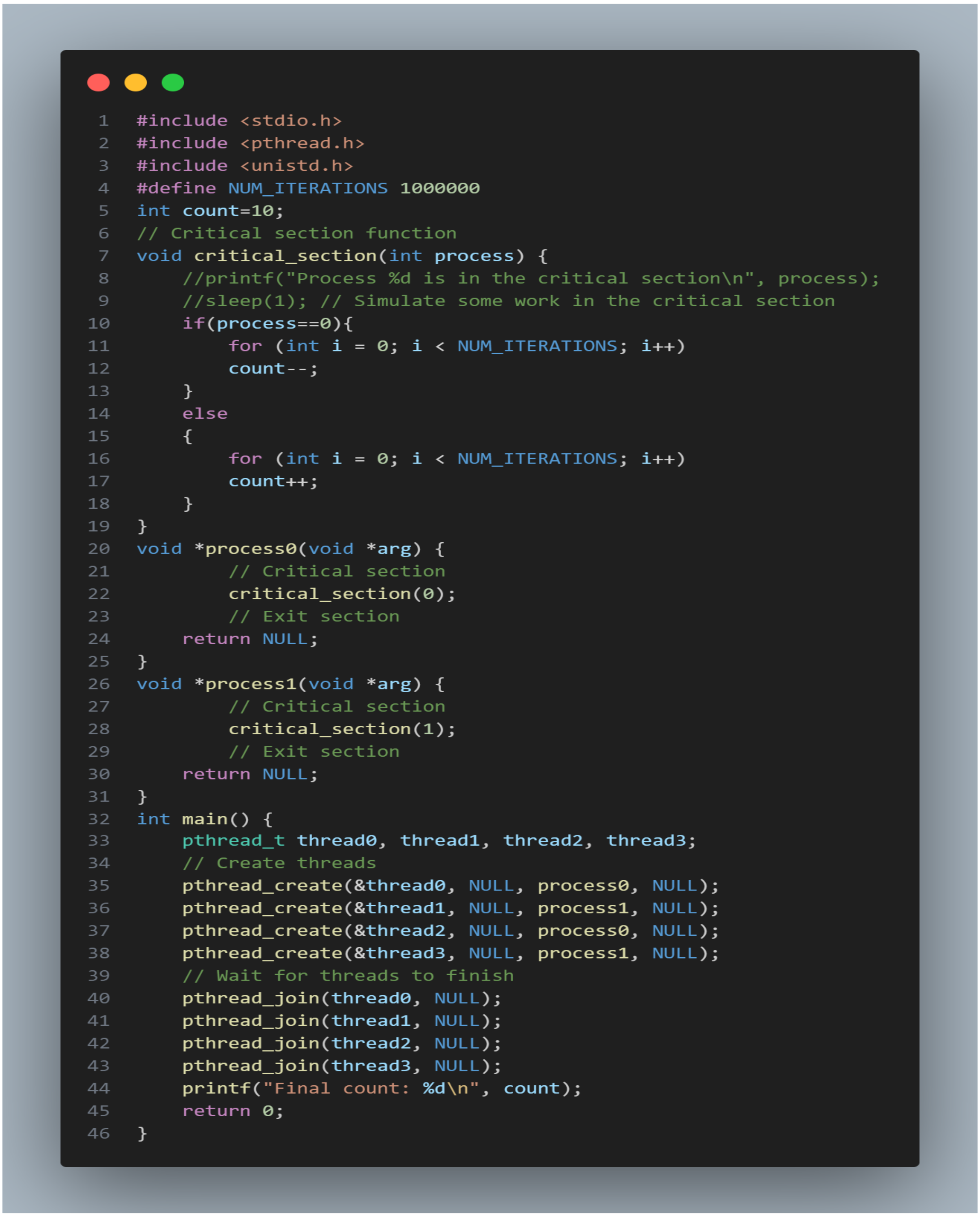


File Edit Selection View Go Run ···
LAB-6-1148 [WSL: Ubuntu-24.04]
EXPLORER
LAB-6-1148 [WSL: UBUNT...
Task1.c U
Task1.out U
Task2.c U
Task1.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - LAB-6-1148
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
main*
Ln 21, Col 1 Spaces: 4 UTF-8 LF {} C Linux
Search
2:45 pm 24/10/2025

```
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task1.c -o Task1.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task1.out
Thread 0 is executing the global value is 1: local vale is 1:   process id 2942:
Thread 1 is executing the global value is 2: local vale is 1:   process id 2942:
Thread 2 is executing the global value is 4: local vale is 1:   process id 2942:
Thread 3 is executing the global value is 4: local vale is 1:   process id 2942:
Main is executing the global value is 4::    Process ID 2942:
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ 
```

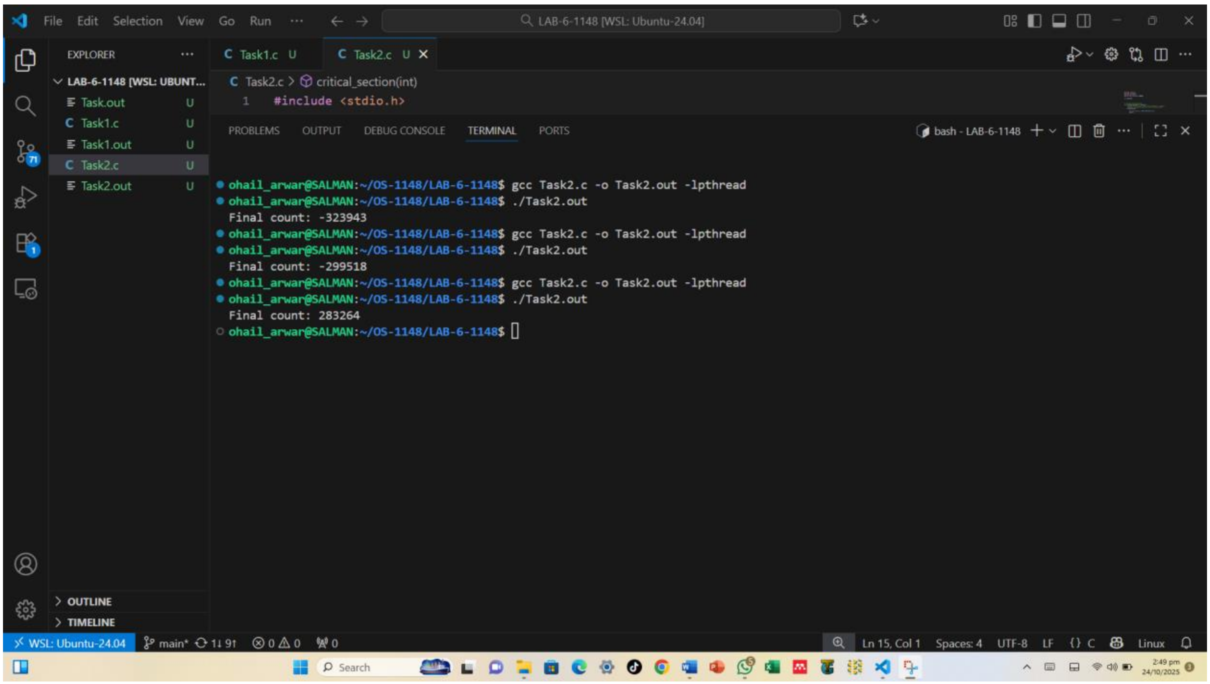
# TASK-2-Critical Section

## Code:



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 1000000
int count=10;
// Critical section function
void critical_section(int process) {
    //printf("Process %d is in the critical section\n", process);
    //sleep(1); // Simulate some work in the critical section
    if(process==0){
        for (int i = 0; i < NUM_ITERATIONS; i++)
        count--;
    }
    else
    {
        for (int i = 0; i < NUM_ITERATIONS; i++)
        count++;
    }
}
void *process0(void *arg) {
        // Critical section
        critical_section(0);
        // Exit section
    return NULL;
}
void *process1(void *arg) {
        // Critical section
        critical_section(1);
        // Exit section
    return NULL;
}
int main() {
    pthread_t thread0, thread1, thread2, thread3;
    // Create threads
    pthread_create(&thread0, NULL, process0, NULL);
    pthread_create(&thread1, NULL, process1, NULL);
    pthread_create(&thread2, NULL, process0, NULL);
    pthread_create(&thread3, NULL, process1, NULL);
    // Wait for threads to finish
    pthread_join(thread0, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_join(thread3, NULL);
    printf("Final count: %d\n", count);
    return 0;
}
```

## Output:



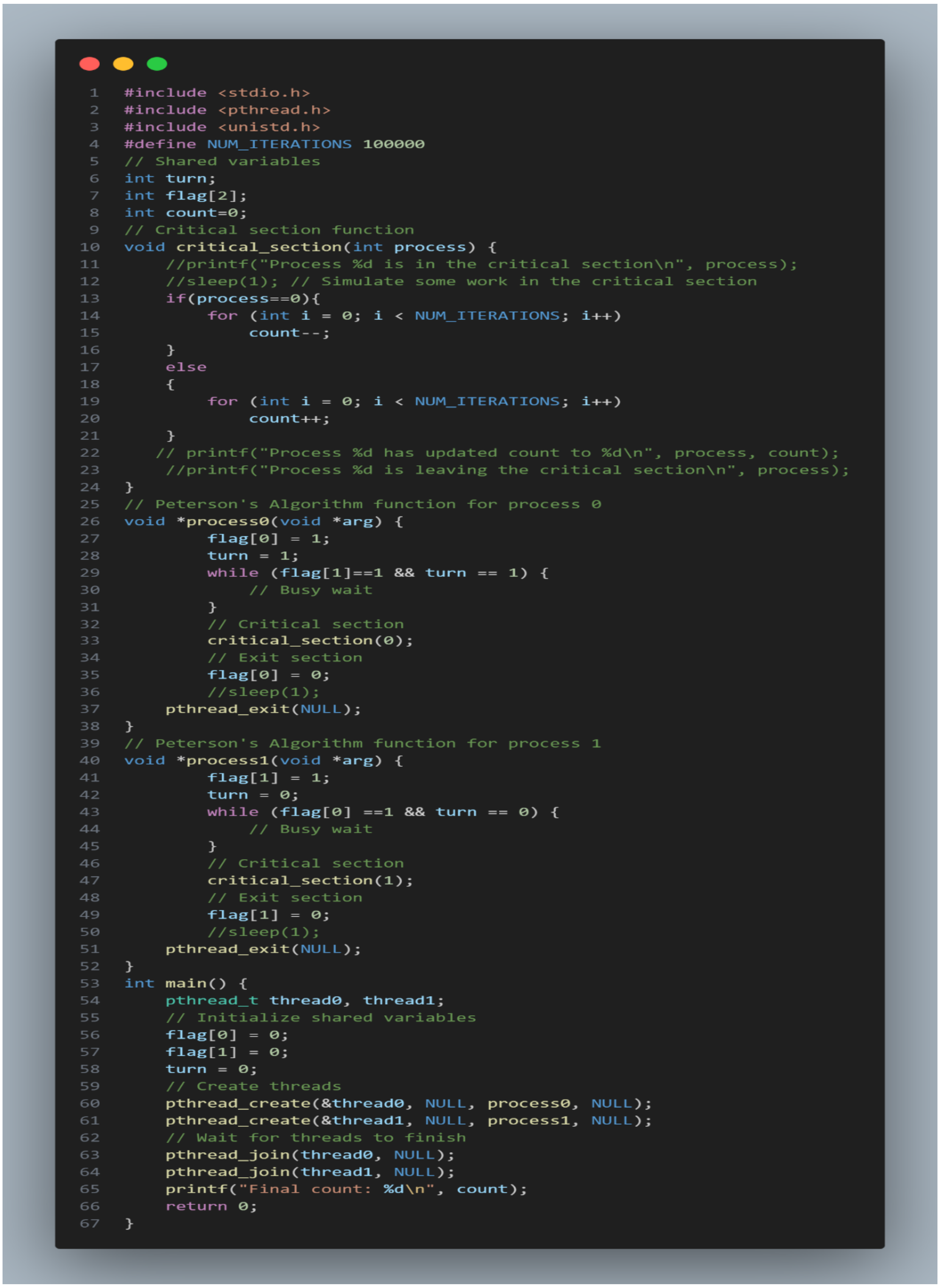
File Edit Selection View Go Run ···
LAB-6-1148 [WSL: Ubuntu-24.04]
EXPLORER
LAB-6-1148 [WSL: UBUNT...
Task.out U
Task1.c U
Task1.out U
Task2.c U
Task2.out U
Task1.c U
Task2.c U
Task2.c > critical_section(int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - LAB-6-1148
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
main*
Ln 15, Col 1 Spaces: 4 UTF-8 LF {} C Linux
Search

```
1    #include <stdio.h>
```

```
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task2.c -o Task2.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task2.out
Final count: -323943
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task2.c -o Task2.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task2.out
Final count: -299518
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task2.c -o Task2.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task2.out
Final count: 283264
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ 
```

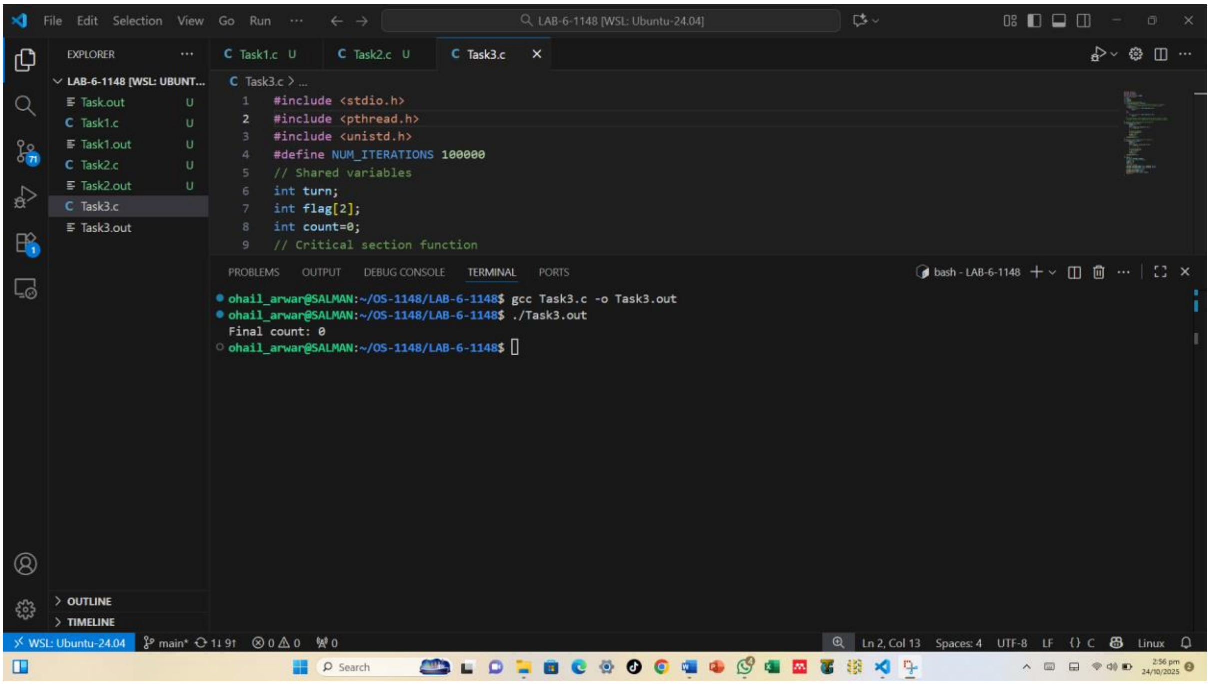
# TASK-3-Peterson

## Code:



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 100000
// Shared variables
int turn;
int flag[2];
int count=0;
// Critical section function
void critical_section(int process) {
    //printf("Process %d is in the critical section\n", process);
    //sleep(1); // Simulate some work in the critical section
    if(process==0){
        for (int i = 0; i < NUM_ITERATIONS; i++)
            count--;
    }
    else
    {
        for (int i = 0; i < NUM_ITERATIONS; i++)
            count++;
    }
   // printf("Process %d has updated count to %d\n", process, count);
    //printf("Process %d is leaving the critical section\n", process);
}
// Peterson's Algorithm function for process 0
void *process0(void *arg) {
        flag[0] = 1;
        turn = 1;
        while (flag[1]==1 && turn == 1) {
            // Busy wait
        }
        // Critical section
        critical_section(0);
        // Exit section
        flag[0] = 0;
        //sleep(1);
    pthread_exit(NULL);
}
// Peterson's Algorithm function for process 1
void *process1(void *arg) {
        flag[1] = 1;
        turn = 0;
        while (flag[0] ==1 && turn == 0) {
            // Busy wait
        }
        // Critical section
        critical_section(1);
        // Exit section
        flag[1] = 0;
        //sleep(1);
    pthread_exit(NULL);
}
int main() {
    pthread_t thread0, thread1;
    // Initialize shared variables
    flag[0] = 0;
    flag[1] = 0;
    turn = 0;
    // Create threads
    pthread_create(&thread0, NULL, process0, NULL);
    pthread_create(&thread1, NULL, process1, NULL);
    // Wait for threads to finish
    pthread_join(thread0, NULL);
    pthread_join(thread1, NULL);
    printf("Final count: %d\n", count);
    return 0;
}
```

**Output:**

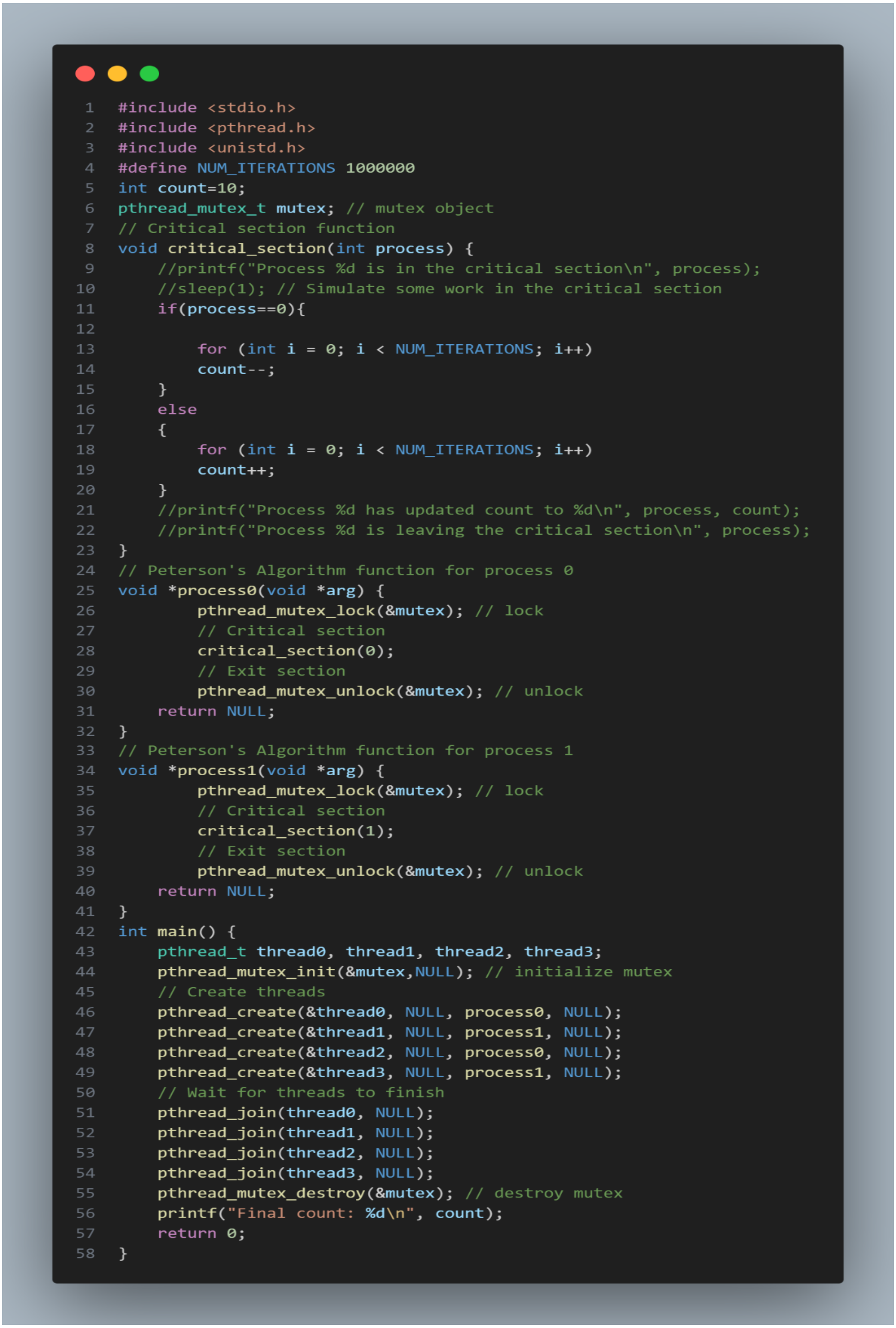


```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 100000
// Shared variables
int turn;
int flag[2];
int count=0;
// Critical section function
```

```
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task3.c -o Task3.out
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task3.out
Final count: 0
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$
```

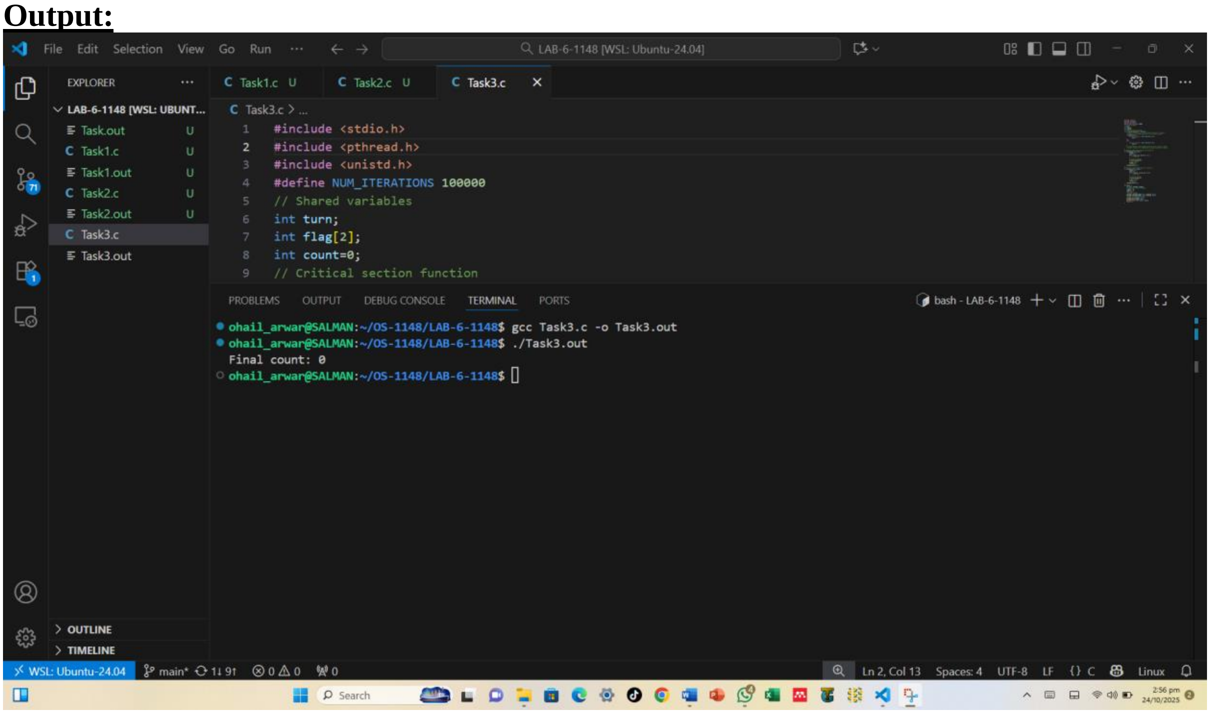
# TASK-4-Mutex

## Code:



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 1000000
int count=10;
pthread_mutex_t mutex; // mutex object
// Critical section function
void critical_section(int process) {
    //printf("Process %d is in the critical section\n", process);
    //sleep(1); // Simulate some work in the critical section
    if(process==0){

        for (int i = 0; i < NUM_ITERATIONS; i++)
        count--;
    }
    else
    {
        for (int i = 0; i < NUM_ITERATIONS; i++)
        count++;
    }
    //printf("Process %d has updated count to %d\n", process, count);
    //printf("Process %d is leaving the critical section\n", process);
}
// Peterson's Algorithm function for process 0
void *process0(void *arg) {
        pthread_mutex_lock(&mutex); // lock
        // Critical section
        critical_section(0);
        // Exit section
        pthread_mutex_unlock(&mutex); // unlock
    return NULL;
}
// Peterson's Algorithm function for process 1
void *process1(void *arg) {
        pthread_mutex_lock(&mutex); // lock
        // Critical section
        critical_section(1);
        // Exit section
        pthread_mutex_unlock(&mutex); // unlock
    return NULL;
}
int main() {
    pthread_t thread0, thread1, thread2, thread3;
    pthread_mutex_init(&mutex,NULL); // initialize mutex
    // Create threads
    pthread_create(&thread0, NULL, process0, NULL);
    pthread_create(&thread1, NULL, process1, NULL);
    pthread_create(&thread2, NULL, process0, NULL);
    pthread_create(&thread3, NULL, process1, NULL);
    // Wait for threads to finish
    pthread_join(thread0, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_join(thread3, NULL);
    pthread_mutex_destroy(&mutex); // destroy mutex
    printf("Final count: %d\n", count);
    return 0;
}
```

## Output:



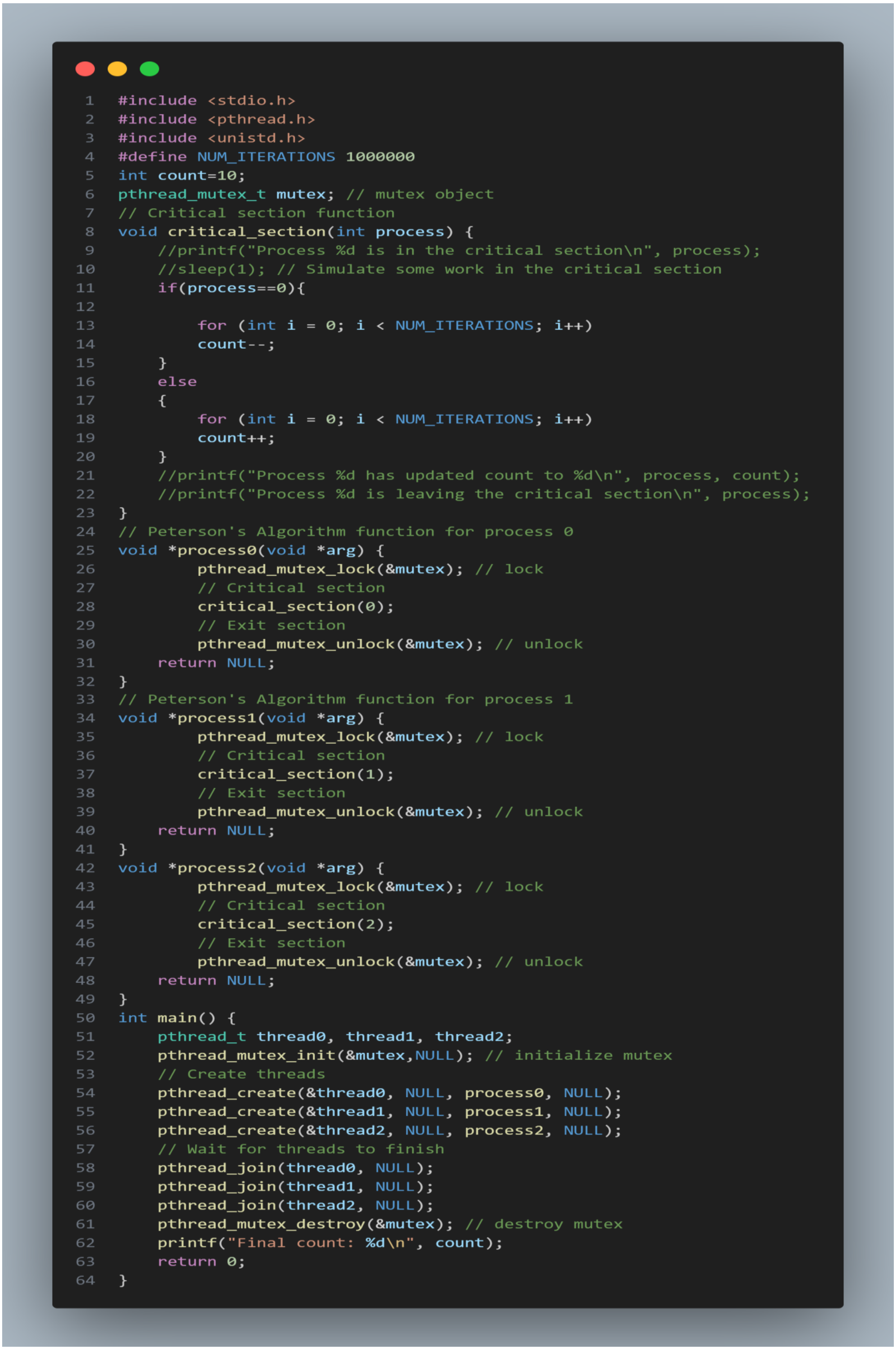
File Edit Selection View Go Run
LAB-6-1148 [WSL: Ubuntu-24.04]
EXPLORER
LAB-6-1148 [WSL: UBUNT...
Task.out U
Task1.c U
Task1.out U
Task2.c U
Task2.out U
Task3.c
Task3.out
Task1.c U
Task2.c U
Task3.c
Task3.c > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - LAB-6-1148
OUTLINE
TIMELINE
WSL: Ubuntu-24.04
main*
Ln 2, Col 13 Spaces: 4 UTF-8 LF {} C Linux

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 100000
// Shared variables
int turn;
int flag[2];
int count=0;
// Critical section function
```

```
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task3.c -o Task3.out
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task3.out
Final count: 0
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$
```

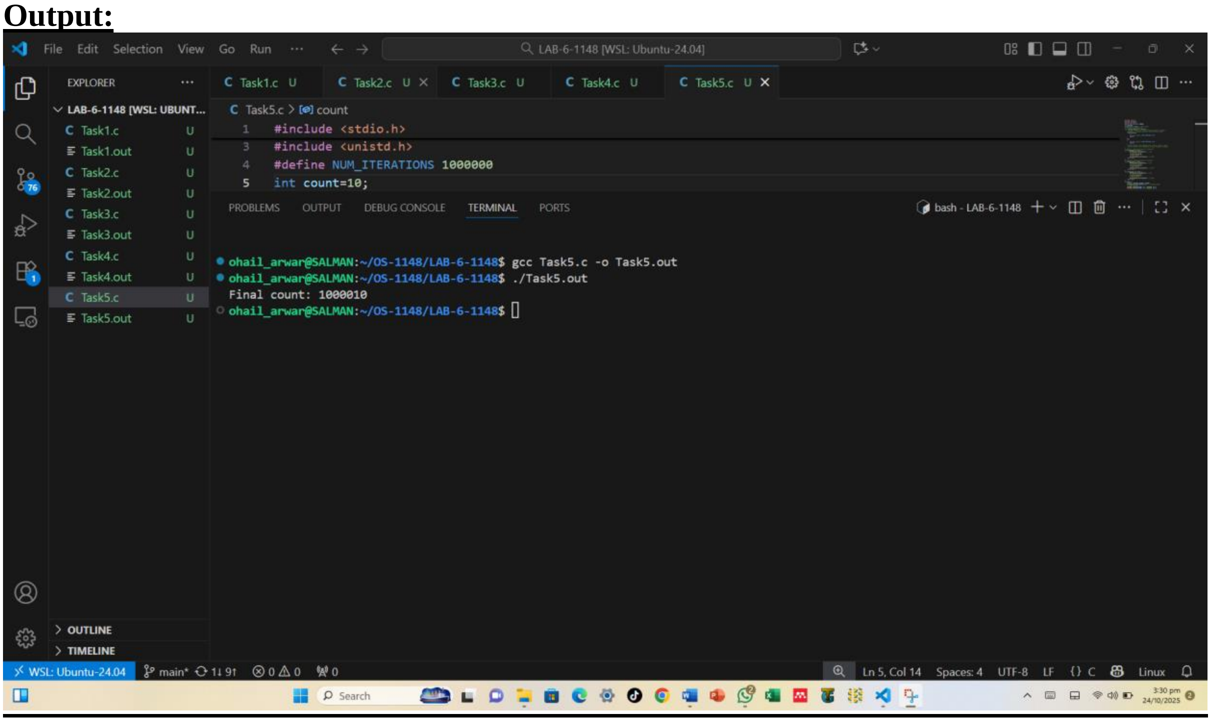
# TASK-5

## Code:



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_ITERATIONS 1000000
int count=10;
pthread_mutex_t mutex; // mutex object
// Critical section function
void critical_section(int process) {
    //printf("Process %d is in the critical section\n", process);
    //sleep(1); // Simulate some work in the critical section
    if(process==0){

        for (int i = 0; i < NUM_ITERATIONS; i++)
        count--;
    }
    else
    {
        for (int i = 0; i < NUM_ITERATIONS; i++)
        count++;
    }
    //printf("Process %d has updated count to %d\n", process, count);
    //printf("Process %d is leaving the critical section\n", process);
}
// Peterson's Algorithm function for process 0
void *process0(void *arg) {
        pthread_mutex_lock(&mutex); // lock
        // Critical section
        critical_section(0);
        // Exit section
        pthread_mutex_unlock(&mutex); // unlock
    return NULL;
}
// Peterson's Algorithm function for process 1
void *process1(void *arg) {
        pthread_mutex_lock(&mutex); // lock
        // Critical section
        critical_section(1);
        // Exit section
        pthread_mutex_unlock(&mutex); // unlock
    return NULL;
}
void *process2(void *arg) {
        pthread_mutex_lock(&mutex); // lock
        // Critical section
        critical_section(2);
        // Exit section
        pthread_mutex_unlock(&mutex); // unlock
    return NULL;
}
int main() {
    pthread_t thread0, thread1, thread2;
    pthread_mutex_init(&mutex,NULL); // initialize mutex
    // Create threads
    pthread_create(&thread0, NULL, process0, NULL);
    pthread_create(&thread1, NULL, process1, NULL);
    pthread_create(&thread2, NULL, process2, NULL);
    // Wait for threads to finish
    pthread_join(thread0, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    pthread_mutex_destroy(&mutex); // destroy mutex
    printf("Final count: %d\n", count);
    return 0;
}
```

**Output:**



```
#include <stdio.h>
#include <unistd.h>
#define NUM_ITERATIONS 1000000
int count=10;
```

```
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ gcc Task5.c -o Task5.out
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ ./Task5.out
Final count: 1000010
ohail_arwar@SALMAN:~/OS-1148/LAB-6-1148$ 
```

| **Peterson** | **Mutex** |
|---|---|
| It is software-based synchronization | it is hardware-based synchronization. |
| Achieving mutual exclusion between two processes only. | Achieving mutual exclusion between two or more threads/processes. |
| It uses two shared variables i.e., flag[] and turn. | It uses system calls like lock() and unlock(). |
| It supported only two processes. | Multiple threads and processes easily supported. |
| It works only on sequentially consistent memory systems. | It uses hardware level atomic operations. |
| Example: flag [i] = true; turn = j;<br>While(flag [j] && turn == j); | Example: pthread_mutex_lock(&lock)…<br>Pthread_mutex_unlock(&lock)… |